# Reconstruction locale et visualisation de nuages de points par surfaces de subdivision

Tamy Boubekeur, Patrick Reuter, Christophe Schlick

# Reconstruction locale et visualisation de nuages de points par surfaces de subdivision

**Tamy Boubekeur, Patrick Reuter, Christophe Schlick**

LaBRI - INRIA Futurs / Université de Bordeaux
boubek|preuter|schlick@labri.fr

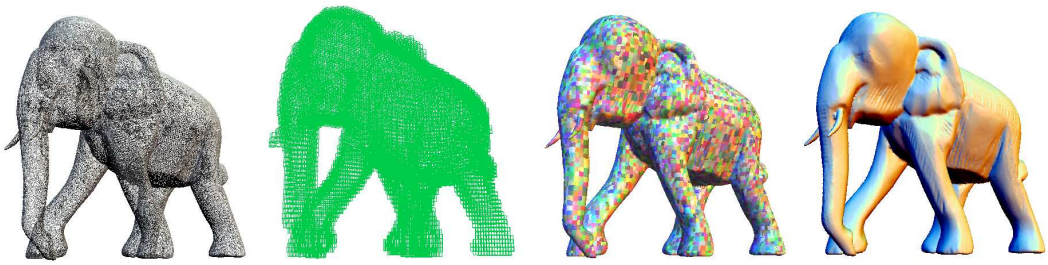

FIG. 1 – Les étapes de notre reconstruction locale d'un nuage de points. L'objet final est un agrégat de surfaces de subdivision indépendantes, sans continuité géométrique, mais qui peut être généré très rapidement et fournit une visualisation lisse très convaincante.

**Résumé :** *Les surfaces de points, qu'elles soient directement acquises par scanner ou issues de la conversion d'autres modèles, permettent de stocker et de transmettre des objets complexes de manière économique, mais sont mal adaptées aux architectures matérielles existantes qui s'appuient sur une description géométrique à base de surfaces polygonales. Cet article propose une technique permettant d'obtenir une visualisation efficace des surfaces de points, entièrement à l'aide du pipeline de rendu matériel. L'idée centrale présentée ici est d'effectuer une reconstruction surfacique locale par triangulation d'un nuage de points, en générant un agrégat de 2-variétés se recouvrant, et en procurant une continuité visuelle dans les zones de recouvrements à l'aide des surfaces de subdivision.*

**Mots-clés :** modélisation géométrique, nuages de points, reconstruction de surface, subdivision de surface, visualisation

## 1 Introduction

### 1.1 Motivations

Dans le cadre de la modélisation géométrique 3D, les *surfaces de points* permettent de représenter des surfaces complexes uniquement par leur composante géométrique — un nuage de points possiblement équipés de normales — sans représentation explicite de la topologie. De nombreuses techniques permettent de passer d'un modèle par points à un modèle continu. On parle dans ce cas de *reconstruction* de surface. La surface obtenue peut être *explicite*, tels que les maillages polygonaux ou les surfaces NURBS, ou bien implicite, c'est-à-dire définie par une fonction $f : R^n \rightarrow R$ avec $n$ étant la dimension de l'espace. La surface est dans ce cas définie comme l'ensemble des points $x$ de $R^n$ tels que $f(x) = 0$. Les travaux sur la *partition de l'unité* [TRS04] montrent qu'il est possible de localiser la reconstruction d'une surface implicite, et de traiter des modèles très volumineux, tels que ceux du projet *Digital Michelangelo* [LPC+00]. Mais les temps de génération sont longs, et les surfaces résultantes, implicites, ne sont pas visualisables directement à l'aide du pipeline graphique standard existant dans les cartes 3D actuelles. A l'opposé, les surfaces polygonales, reconstruites explicitement, offrent cette visualisation efficace, mais ne proposent pas de véritable approche localisée — similaire à la partition de l'unité — permettant de ne reconstruire que les zones d'intérêt, en tirant partie des informations de visibilité par exemple, ou bien encore de paralléliser la reconstruction. On propose dans cet article un nouvel algorithme de reconstruction de surfaces issues de points, notamment destiné à la visualisation rapide. Notre approche est locale, et s'appuie sur les surfaces de

subdivision pour la grande variété de surfaces qu'elles peuvent exprimer (surfaces lisses par morceaux, à bords, arêtes vives), et leur adéquation avec le pipeline de rendu matériel (génération d'un ensemble de polygones).

## 1.2 Travaux précédents

Les surfaces de points [Kri03, ZPBG01, ZPKG02] sont considérées comme un mode de représentation majeur de surface 3D, pouvant servir de noeud central entre tous les autres à travers des techniques de reconstruction. Les initiateurs de ce domaine de recherche furent Hoppe et al. [HDD+92, HDD+93, HDD+94] qui proposèrent une approche en 3 étapes : une extraction d'un maillage initial à partir d'une fonction de distance construite sur les points, suivie d'une optimisation de ce maillage et enfin d'une application d'un schéma de subdivision. Depuis ces travaux fondateurs, de nombreuses solutions au problème de la reconstruction ont été proposées. La classification de ces différentes techniques laisse apparaître deux grandes familles : reconstruction implicite ou explicite.

**Reconstruction Implicite :** Il existe une première approche dans ce cadre qui consiste à utiliser des *fonctions à base radiale* (RBF) pour interpoler un nuage de points [TO02]. Initialement limitée à des nuages de points de petite taille, cette approche a par la suite été étendue aux nuages de points de grande taille via le principe de la *partition de l'unité* [TRS04], technique qui a également été utilisée sur d'autres types de surfaces implicites [OBA+03]. Une seconde approche possible est d'utiliser les surfaces *MLS* [ABCO+01] sur un nuage de points, à l'aide d'un opérateur de projection. Le point fort des méthodes de reconstruction implicite est d'être très robustes lorsque le nuage de points n'est pas uniforme. Leur point faible, comme nous l'avons déjà indiqué, est que la surface résultante n'est pas directement visualisable par le pipeline de rendu matériel et nécessite une étape supplémentaire d'extraction de maillage de la surface implicite de type *Marching Cubes* [Blo88, LC87].

**Reconstruction Explicite :** Là encore, il existe globalement deux approches pour générer une reconstruction polygonale du nuage de points. D'une part une approche dynamique, dérivée des *contours actifs* [BI98, KWT87] issues de la 2D, où l'on essaie de minimiser une énergie [Alg95] [DYQS04] définie par le nuage de points, en tout point d'une surface graine, et ce en déformant cette surface ; on peut citer notamment les *Intelligent Balloons* [DQ01] pour leur flexibilité topologique. Ces techniques sont très lentes, et peu adaptées aux grands nuages de points. D'autre part, une approche statique, basée sur une triangulation, habituellement celle de Delaunay [BC00, CSD02] et son dual topologique, le diagramme de Voronoï. La technique du *Crust* d'Amenta et al. [ACK01] s'appuie sur ce diagramme, et extrait une surface à partir d'une analyse de l'axe médian ; cet algorithme fut optimisé par le *Cocoone* de Dey et al. [DGH01]. Pour gérer des nuages de points plus importants, des approches par projections locales furent proposées par Gopi et al. [GKS00, GK00]. Outre l'ajout de qualité à la surface en post-traitement, la subdivision de surface [ZS00] est en général utilisée en reconstruction après la génération d'un maillage de faible définition à partir du nuage de points [JK02, HSK99]. L'application de passes de subdivision successives, en minimisant l'erreur avec le nuage de points à chaque étape converge vers une surface approximant le nuage de points. De nombreuses implémentations des surfaces de subdivision ont été proposées [BS02, BS03, Bri02, VPBM01], ainsi que des panels complets de représentations de surfaces quelconques par surfaces de subdivision [BLZ00, HBB01].

Il faut noter que nous n'avons parlé que des reconstructions explicites s'effectuant dans l'espace objet, qui seules permettent d'obtenir des surfaces prises en charge directement par le pipeline matériel. Les techniques de reconstruction dans l'espace image — le *rendu par points* [ZPBG01, BSK04] — nous semblent peu adaptées aux grandes résolutions d'images, limitées en terme d'apparence, et nécessitent des modifications spécifiques du pipeline de rendu pour être efficaces.

## 1.3 Vue d'ensemble

Notre approche — une reconstruction de surface explicite conçue pour la visualisation — est de séparer clairement la reconstitution de la topologie globale du recouvrement local de la surface. L'algorithme prend en entrée un nuage de points non structuré, où les points sont équipés d'une normale, et fournit en sortie un ensemble disjoint de 2-variétés à bord, toutes de genre 0. Le processus d'agrégation de ces 2-variétés permet d'obtenir une surface visuellement continue (voir figure 2). Afin de traiter les gros nuages de points, on adopte une approche locale. Toute opération de comparaison, test, tri, ou bien encore sélection ne se fait ainsi que sur un ensemble réduit

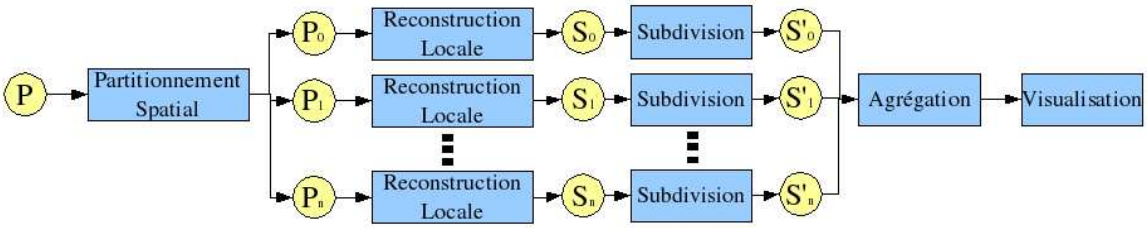


FIG. 2 – Principe général de reconstruction locale.

de points à un moment donné. Une solution comparable à la *partition de l'unité* dans le cas de la reconstruction implicite [TRS04, OBA+03] est adoptée. Cette solution permet également d'envisager de paralléliser l'algorithme, d'effectuer une reconstruction hors-mémoire [Sil03] ou bien encore d'effectuer une reconstruction en fonction de différents critères (voir section 5). L'algorithme présenté ici peut se décomposer en 3 étapes :

1. Partitionnement en octree du nuage de points (section 3).
2. Reconstruction locale de chaque partition (section 2), en l'augmentant momentanément des points de son voisinage. L'objectif étant d'éliminer les trous en créant des *recouvrements* entre surfaces voisines.
3. Application du principe de subdivision à l'agrégat de surfaces obtenu, afin d'augmenter la qualité visuelle de l'ensemble (section 4).

**Notation :** Soit $P$ l'ensemble de points d'entrée, on note $P_i$ la $i$ème partition de cet ensemble, tel que $\bigcup_i P_i = P$, avec les ensembles $P_i$ disjoints. $C_i$ représente la cellule associée à la partition $P_i$ (ie le plus petit volume issu de l'octree contenant $P_i$), $N_i$ le nombre de points contenus dans $C_i$. Pour chaque partition $P_i$, on note $S_i$ le morceau de surface reconstruit sur la partition $P_i$. On note $V_i$ l'ensemble des partitions voisines de $P_i$. On note $p_{ij}$ le jème point de la partition $P_i$. On note $c_i$ le barycentre de $P_i$ et $s_i$ le rayon de $C_i$ (la moitié du plus grand axe de $C_i$). Enfin, on note $\overrightarrow{n_{ij}}$ la normale du $j$ème point de $P_i$ et $\overrightarrow{n_i}$ l'interpolation entre les normales de $P_i$.

# 2 Reconstruction

## 2.1 Reconstruction surfacique locale

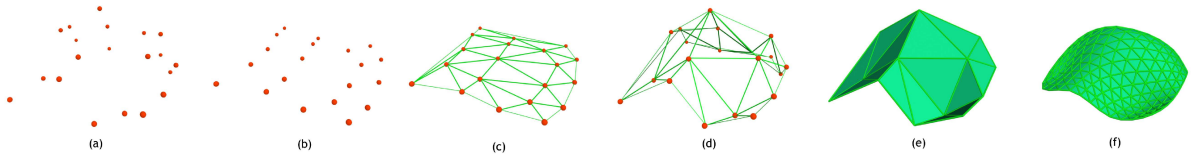


FIG. 3 – Reconstruction surfacique locale. (a) Partition initiale $P_i$ (b) Projection sur le plan minimisant l'énergie pour $P_i$. (c) Reconstruction en 2D par triangulation de Delaunay. (d) Reprojection 3D de l'ensemble des triangles obtenus. (e) Surface reconstruite $S_i$ avant subdivision. (f) Surface lisse obtenue par subdivision de $S_i$.

L'algorithme de reconstruction locale doit avoir un certain nombre de propriétés :
– prendre en entrée un petit nuage de points non organisé $P_i$ et fournir en sortie un morceau de surface $S_i$ ;
– la surface reconstruite $S_i$ doit interpoler, ou à défaut fournir une bonne approximation de $P_i$ ;
– $S_i$ sera soumis à la subdivision, et doit donc le plus possible se rapprocher des critères de qualité de subdivision (valence des sommets [ZS00])).
On ne fera pas de différence forte entre *point* et *surfel* (*surface element*, c'est-à-dire un point équipé d'une normale), en considérant que l'on peut passer de l'un à l'autre par une analyse en composante principale du voisinage [HDD+92, GKS00]. Le coeur de l'algorithme de reconstruction se base sur la *triangulation de Delaunay*, principalement choisie pour ses bonnes propriétés (valence relativement régulière des sommets, maximisation de l'angle

minimal). Un des principaux reproches habituellement fait à cette triangulation, dans le cadre d'une surface 3D, est sa complexité : en effet, une triangulation de Delaunay 3D demande, après avoir généré une 3-variété composée de tétraèdres interpolant les points, de sélectionner uniquement les triangles situés sur la surface, ce qui n'est pas forcément simple et surtout, implique de devoir "jeter" une grande partie des informations calculées — tous les triangles intérieurs du volume — car c'est bien un objet en 2 dimensions, la surface, que l'on essaie ici de générer. En fait, si l'on considère le problème localement, il n'y aucune notion de solide 3D et dans cet esprit, on peut considérer la surface comme une carte d'élévation locale (*local height map*) par rapport au plan tangent moyen. Autrement dit, la surface va pouvoir s'exprimer comme une fonction qui à chaque point du plan tangent moyen va associer une élévation le long de la normale. Dès lors, on peut utiliser une triangulation 2D qui génère une 2-variété de triangles interpolant le nuage de points. Pour que notre hypothèse de travail fonctionne, il faut s'assurer que le nuage de points $P_i$ puisse effectivement être considéré comme une carte d'élévation. Pour cela on définit un prédicat $\kappa$, assurant qu'un ensemble de points équipés de normales est une *surface d'élévation* :

$$\forall k \in [0, N_i[,\ \overrightarrow{n_{i,k}}.\overrightarrow{n_i} > \delta_a,\ \delta_a \in [0,1] \text{ et } ||\overrightarrow{p_{i,k}, c_i}|| < \delta_d,\ \delta_d \in [0, s_i] \Rightarrow \kappa(P_i) \text{ est vrai}$$

Le paramètre $\delta_a$ est lié à l'angle maximal que peut former la normale en un point avec la normale moyenne d'une surface d'élévation. Les tests montrent que $\delta_a = 0.1$ donne de bons résultats en général. Le paramètre $\delta_d$ représente un écart-type maximal autorisé pour la distribution des points autour de $P_i$ (on peut se contenter d'effectuer les tests sur la variance, ce qui évite les calculs de racines carrées). En l'absence de critère de densité $\gamma$, on choisit $\delta_d = \frac{s_i}{4}$ qui donne de bons résultats. Ce paramètre est assez intuitif pour être modifié interactivement et limite les dégénérescences dues au partitionnement. Cette approche n'est que sub-optimale, dans la mesure où $P_i$ est déterminé par le partitionnement en octree (section 3). En particulier, on ne maximise pas l'aire des surfaces d'élévation. On choisit cependant cette formulation pour son coût d'implémentation faible. L'un des avantages d'avoir un double critère qui agit à la fois sur les variations des normales et sur celles des positions est que l'ensemble des cas "difficiles" sont détectés correctement (repliement local de surface, composantes non-connexes, etc).

Si $\kappa(P_i)$ est vrai pour tout $i$, on peut raisonner en 2D, et reconstruire la collection de surfaces à l'aide de triangulations 2D. Gopi et al. [GKS00] avaient exploré cette idée, mais en considérant les points de manière isolée. On propose ici une approche différente, en élargissant cette reconstruction de dimension inférieure à toute une partie de la surface, et en garantissant un accès complètement local aux données. L'algorithme pour chaque cellule $C_i$ est le suivant :

1. Calculer le plan tangent moyen $\Pi_i$ (qui va minimiser l'énergie dans $C_i$) ayant pour normale $n_i$ et passant par $c_i$ ;
2. Projeter $P_i$ sur le plan $\Pi_i$ ;
3. Calculer la triangulation de Delaunay en 2D sur les points projetés sur $\Pi_i$ ;
4. Reprojeter les triangles obtenus dans l'espace 3D par transformation inverse de l'étape 2.

La figure 3 montre les différentes phases de cet algorithme. Cette solution interpole $P_i$ dans chaque cellule $C_i$, et fournit une bonne distribution des triangles. On génère beaucoup moins de triangles qu'une triangulation 3D, et les calculs à effectuer sont plus simples (géométrie du plan). Cette triangulation offre l'avantage de maximiser le plus petit angle formé par 2 arêtes. La valence aux sommets extraordinaires est donc faible, et l'application de plusieurs passes de subdivision ne verra pas se développer de singularités trop fortes en ces points. Bien entendu, l'étape de reprojection 3D occasionnera une déformation des triangles, mais les tests montrent que le maillage reste correct pour les densités courantes. Cet algorithme s'est montré très robuste face à une grande variété de modèles de tests.

## 2.2 Recouvrements

Afin d'assurer une visualisation correcte — sans trou entre morceaux de surface — on propose d'effectuer un recouvrement entre surfaces voisines, en appliquant l'opérateur de reconstruction locale sur des partitions momentanément élargies à leur voisinage (figure 5). Cette étape s'apparente à une restitution des caractéristiques globales de la topologie. On élargit le support de reconstruction, sans contrainte particulière, un peu à la manière de la *partition de l'unité* pour la reconstruction implicite (section 1.2). Chaque cellule est élargie d'un facteur $\beta$, afin d'englober les points voisins. Cette technique provoque un recouvrement entre morceaux de surface, et élimine ainsi les trous. Déterminer $\beta$ peut se faire interactivement, ou bien en tenant compte de la densité du nuage de points. On parle de $\gamma$-densité d'un nuage de points si toute boule centrée en un point du nuage, de rayon $\gamma$ ne

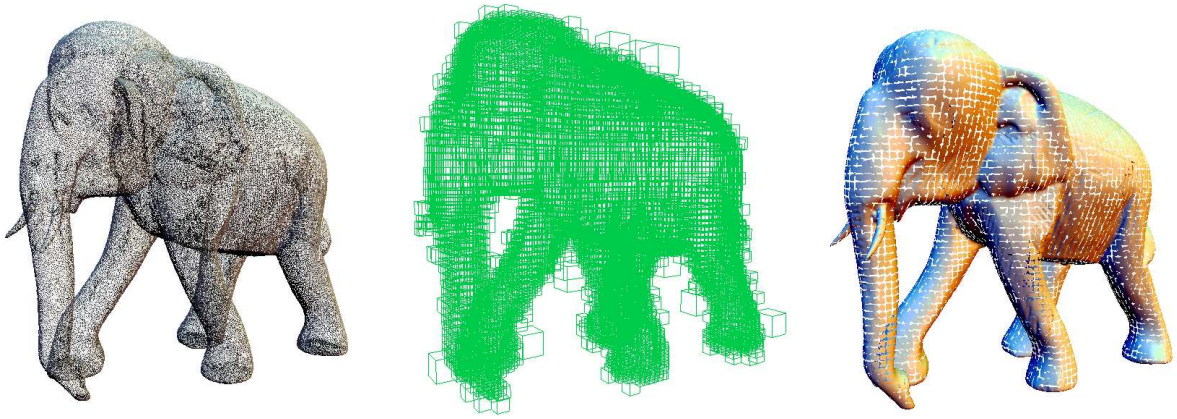

FIG. 4 – A gauche, le nuage de points $P$. Au centre, on visualise les cellules feuilles de l'octree partitionnant le nuage. A droite, un morceau de surface a été reconstruit indépendamment pour chaque partition.

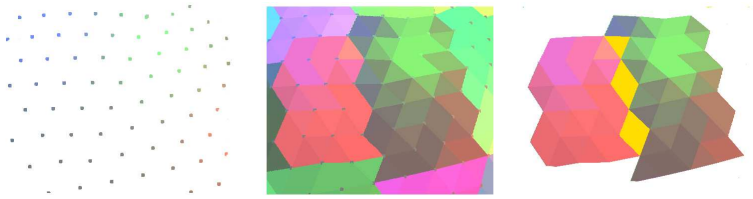

FIG. 5 – A gauche une partie d'un nuage de points ; au centre, la collection de surfaces reconstruites sur ces points ; à droite, en jaune, la zone de recouvrement entre 2 surfaces.

contient aucun autre point, et que $\gamma$ est maximal. Dans ces conditions, il suffit d'élargir le support de $\beta = \frac{2\gamma}{s_i} + 1$ pour garantir un recouvrement suffisant. Malheureusement, l'information de densité n'est pas toujours disponible. On peut dans ce cas se baser sur la taille des cellules : dans chaque cellule $C_i$, l'opérateur de reconstruction locale est appliqué sur $Q_i = P_i \cup R_i$ où $R_i$ est l'ensemble défini par : $p \in R_i \iff p \in V_i$ et $||\overrightarrow{pc_i}|| < \beta.s_i$. Expérimentalement, $\beta = 1.5$ donne de bons résultats sur l'ensemble des nuages de points testés avec l'algorithme. Attention, il faut préserver la propriété de surface d'élévation lors de l'élargissement de l'ensemble de points, et donc vérifier la validité de $\kappa(Q_i)$. L'opérateur de reconstruction locale étant interpolant, la contrainte géométrique est très forte : les morceaux de surface ne peuvent avoir pour support que les points en entrée. Or la triangulation de Delaunay offre toujours un bon équilibre des triangles générés. Ainsi dans la pratique, le recouvrement est soit exact (avec des triangles se superposant parfaitement), soit assez proche d'un recouvrement exact sans dégénérescence forte (figure 6).

## 2.3 Complexité

Etant donné le mécanisme d'élargissement des cellules utilisé par notre algorithme, le nombre de points $n$ servant de base à la triangulation est supérieur au nombre de points initiaux (30% en moyenne). La complexité mémoire de l'algorithme de Delaunay en dimension $d$ est de $\Theta(n^{\lceil \frac{d}{2} \rceil})$, alors qu'elle est de $\Theta(n^{1.8})$ pour les surfaces de points [AB01]. Dans notre cas, la résolution se fait en 2 dimensions : la complexité mémoire est donc linéaire par rapport au nombre de points. Chaque ajout de points à la triangulation demande de tester l'ensemble des triangles déjà générés. On peut pour cela utiliser une structure arborescente, qui induit une complexité calculatoire de l'ordre de $n \log n$ pour l'insertion de $n$ points dans le diagramme. Dans notre approche, les $n$ points sont répartis entre $m$ cellules feuilles d'un octree avec $m < n$. Si l'on majore le nombre de points par cellule à $T$, une constante indépendante de $n$, on a donc $m = \lceil n/T \rceil$ cellules. La complexité calculatoire pour une cellule est de l'ordre de $\Theta(T \log T)$, une constante. La complexité de la triangulation totale est donc de l'ordre de $m.\Theta(T \log T)$, soit $\Theta(n)$. Il faut y ajouter les complexités des opérations de projection et reprojection qui sont également linéaires.

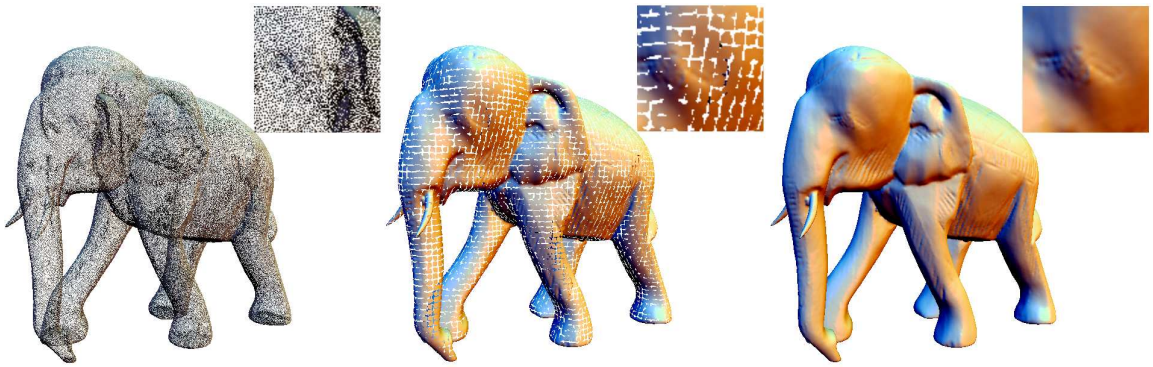

FIG. 6 – De gauche à droite : le nuage de points, l'agrégat de surfaces générées sans recouvrement et avec. Même sous un fort taux de zoom, la continuité visuelle est préservée.

# 3 Partitionnement

## 3.1 Structure de partitionnement

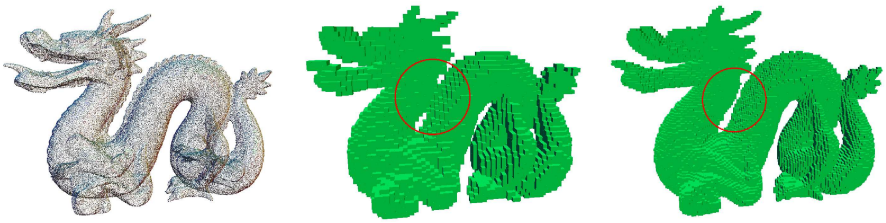

FIG. 7 – A gauche le modèle du dragon avec environ 100 000 points. Au milieu, l'octree avec une profondeur maximale de 5. On observe une erreur de topologie globale dans la zone cerclée de rouge. A droite, on a augmenté la profondeur maximale, et la topologie semble juste : à priori, l'octree est homéomorphe à l'objet échantillonné.

L'approche locale décrite plus haut impose une structure de partitionnement approximant la topologie globale afin de permettre l'élargissement des partitions pour le recouvrement. Parmi les candidats possibles, nous avons choisi l'*octree* pour sa structuration hiérarchique où les cellules ont une forme régulière et l'approximation, géométriquement adaptative, qu'il fournit pour la topologie du nuage de points. En particulier, le genre de la surface va pouvoir être retrouvé facilement (voir figure 7), ce qui évite de s'en préoccuper à l'étape locale. On note de plus que l'octree nous permet le cas échéant d'induire une notion de multirésolution dans la reconstruction de nuages de points. Inversement, parmi les défauts clairement identifiés de l'octree, il faut signaler la dépendance affine de la décomposition (le résultat généré est différent selon la position et l'orientation du cube englobant initial).

## 3.2 Partitionnement en octree

A l'aide du prédicat $\kappa(P)$ défini dans la section 2.1, l'algorithme de partitionnement en octree devient très simple. On part de $C$, cube englobant de $P$. Si $\kappa(P)$ est faux, alors on découpe $C$ en 8 cellules égales, on leur associe les points de $P$ qu'elles contiennent, et on relance le test sur chaque cellule. La figure 8 illustre ce principe en 2D. Afin de préserver une complexité calculatoire acceptable, une cellule $C_i$ vérifiant $\kappa$ sera quand même subdivisée si $N_i$ est trop grand (voir figure 9). A l'usage, on constate qu'un partitionnement fournissant autour de 16 points par cellule donne de bons résultats pour la suite de l'algorithme. Hormis le fait de mieux équilibrer la complexité calculatoire, l'équilibrage du nombre de points par cellule permet également de diminuer la dépendance affine du partitionnement, notamment avec des nuages de points uniformes.

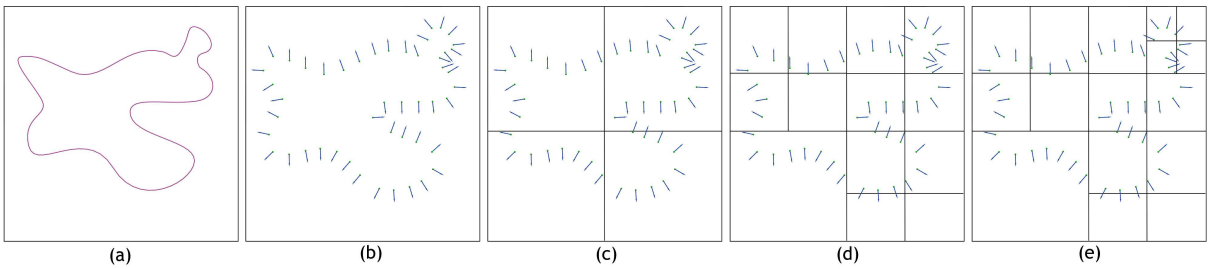


FIG. 8 – Un exemple 2D de partitionnement — un quad-tree se substitue à l'octree — selon le critère de *surface locale d'élévation* $\kappa$. (a) La forme d'origine. (b) Le nuage de points équipés de normales. (c),(d),(e) Raffinements successifs du partitionnement : toute cellule dont le nuage de points ne satisfait pas $\kappa$ est subdivisée.

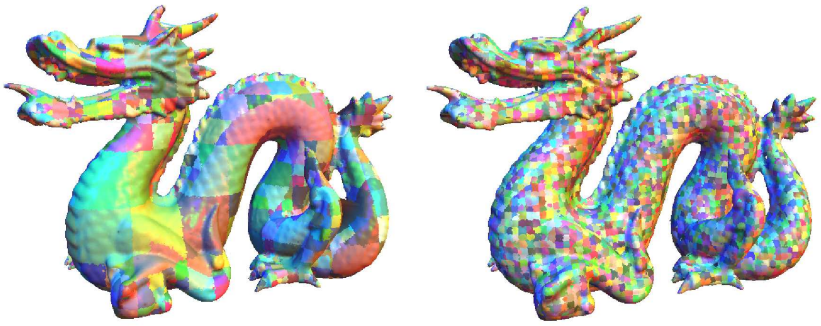

FIG. 9 – A gauche, la reconstruction après un partitionnement adaptatif. A droite, on a ajouté la contrainte du nombre de points par cellule.

### 3.3 Topologie discrète

Afin d'avoir un recouvrement entre surfaces qui respecte la topologie globale de l'objet, on utilise le voisinage volumique fournit par l'octree pour décider quels points viennent augmenter une partition donnée. La régularité de l'octree — et en particulier le fait qu'il soit exclusivement composé de cubes — induit un algorithme assez simple de recherche du voisinage. On rappelle que l'intersection de la fermeture des volumes de 2 cellules voisines est non-vide (une face en 6-connexité, une face ou une arête en 18-connexité, une face, une arête ou un sommet en 26-connexité). Soient deux cellules — cubiques — $A$ et $B$ de l'octree $O$ aligné sur les axes $\{x, y, z\}$. Soient $r_A$ et $u_A$ (respectivement $r_B$ et $u_B$) la longueur d'arête et le centre de $A$ (respectivement $B$). Soit $r_{min}$ la longueur d'arête de la plus petite cellule de $O$. Soit $\zeta_X(\overrightarrow{u})$ la norme Euclidienne de la projection de $\overrightarrow{w}$ sur l'axe $X$. Alors,

$$A \in V_B \text{ ssi } \max(\zeta_x(\overrightarrow{u_A u_B}), \zeta_y(\overrightarrow{u_B - u_B}), \zeta_z(\overrightarrow{u_A u_B})) - r_{min} < \frac{r_A + r_B}{2}.$$

Établir le voisinage de l'ensemble des cellules d'un nuage de 450 000 points demande, avec cette méthode, moins d'une seconde sur une machine de bureau standard, ce qui autorise un réglage interactif des paramètres du partitionnement. On considère ici que la surface cherchée est localisée dans la *couche* volumique approximée par les *feuilles* de l'octree. La notion de voisinage dans cette couche va être une approximation convaincante de la notion de voisinage sur la surface. Autrement dit, un morceau de surface $S_i$ trouvera son voisinage dans les surfaces portées par $V_i$.

## 4 Subdivision

Après les phases de partitionnement (section 3) et de triangulation locale (section 2), nous avons à notre disposition une collection de 2-variétés disjointes mais recouvrantes, chacune d'elles étant définie sous forme d'un

maillage triangulaire composé d'une trentaine de triangles en moyenne. L'étape suivante est d'appliquer le schéma de subdivision de Loop [Loo87, ZS00] à chaque surface $S_i$. Ce schéma garantit une surface limite au moins $C_1$, il est approximant (ce qui a pour effet de lisser le bruit des nuages de points), et il autorise des arêtes vives [HDD+94, BLZ00]. Dans notre algorithme, l'intérêt des surfaces de subdivision va être double.

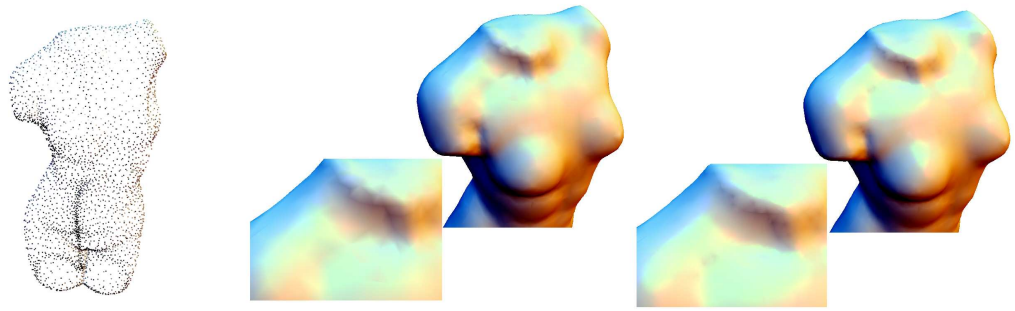

FIG. 10 – A gauche, un nuage de points. Au centre, la collection de surfaces obtenue après la reconstruction locale par recouvrements. A droite, l'application de 2 passes de subdivision sur cette collection produit une visualisation de meilleure qualité avec OpenGL.

D'une part, l'application de quelques étapes de subdivision à l'ensemble des surfaces $S_i$ va permettre d'améliorer la qualité de la surface obtenue :
– tendre vers une surface lisse, et retrouver ainsi l'un des avantages de la reconstruction par surfaces implicites ;
– générer plus de triangles, ce qui augmente la qualité de l'illumination de la surface (voir figure 10) ;
– arrondir les bords ce qui atténue en partie l'effet produit par les recouvrements.
D'autre part, la subdivision va permettre de transformer la collection de 2-variétés en un agrégat, en faisant converger, dans les zones de recouvrement, les morceaux de surface reconstruits localement. En effet, $S_i$ est une solution satisfaisante à la reconstruction dans $C_i$, ce qui n'est pas le cas dans $V_i$ et donc dans les zones de chevauchement entre $S_i$ et ses surfaces voisines. Soit $S_i^R$ la partie de la surface $S_i$ située hors de $C_i$ (zones de recouvrement). On propose de conformer tout point $p_{jk}$ de $S_i^R$ à la surface $S_j$. On effectue cette conformation à travers le processus de subdivision, en ajoutant une phase de projection des points. On répète les phases :

1. Application d'une subdivision.
2. Projection des points hors-cellule $p_{jk}$ vers $S_j$.

Une projection efficace consiste à chercher l'intersection entre le rayon défini par $p_{jk}$ et sa normale associée, et la surface $S_j$. Cette solution ne garantit pas d'intersection, mais s'est montrée satisfaisante en pratique. On peut envisager par la suite de développer un opérateur de projection plus robuste, basé par exemple sur la surface MLS définie par les points de la zone de chevauchement. Mais on peut craindre que cette solution sera vraisemblablement plus lente.

En terme de visualisation, peu d'itérations de ce principe sont nécessaires. On envisage d'accélérer la convergence du processus en projetant systématiquement les points nouvellement créés sur la surface limite issue de la subdivision. Cette phase peut être assimilée à l'interpolation présente dans la partition de l'unité et constitue un élément essentiel de nos travaux en cours.

# 5 Visualisation

## 5.1 Rendu matériel

Comme toutes les méthodes explicites, notre algorithme fournit des surfaces polygonales qui peuvent être traitées par le pipeline de rendu matériel et les moteurs de rendu courants. L'agrégat de surfaces, naturellement organisées en octree, peut tirer partie d'algorithmes de détection d'occlusions [COCS00] pour en accélérer le rendu. Notre implémentation dispose d'un module de rendu OpenGL qui permet de valider l'hypothèse initiale de notre démarche : même si l'agrégat de 2-variétés reconstruites ne fournit pas une continuité au sens géométrique, elle en est suffisamment proche pour que la visualisation ne souffre pas des zones de recouvrement (voir figure 11).

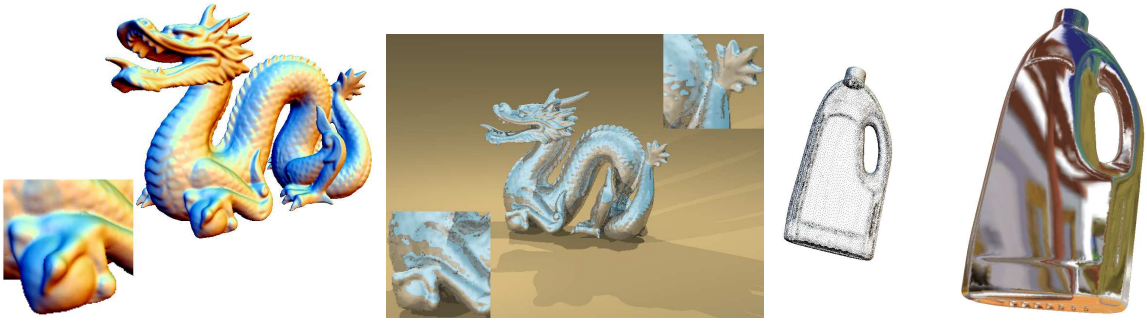

FIG. 11 – Deux visualisations différentes du modèle du Dragon (437 646 points) après reconstruction. A gauche, un rendu en temps réel avec OpenGL. Au centre, un rendu par lancé de rayon avec un shader de réflexion. Les artefacts dûs aux recouvrements sont peu ou pas visibles. A droite, un exemple de bénéfice immédiat de notre technique : l'utilisation directe de *Cube Mapping* en OpenGL.

### 5.2 Visualisation de points par lancé de rayons

Supposons que l'on souhaite visualiser un nuage de points par lancé de rayons, une technique qui recommence à susciter l'intérêt des développeurs grâce aux nouvelles possibilités des GPU [DWBS03]. Une approche naïve pourrait être de reconstruire la surface à l'aide de l'algorithme proposé dans cet article, puis de la soumettre au lancé de rayons. Cependant, il est probable pour une image fixe, voire pour une animation, qu'une partie de la surface soit cachée sur l'ensemble de la séquence. On aura alors inutilement allongé le temps de reconstruction pour des morceaux de surface qui n'apparaîtront pas. C'est un problème de *visibilité*, et on trouvera dans l'article de Cohen-Or et al. [COCS00] un tour d'horizon des techniques récentes pour pallier à ce problème. On propose ici d'effectuer la reconstruction *à la volée* en tirant partie de notre approche locale. Seules les cellules de l'octree intersectées par un rayon seront reconstruites. Le cas de figure où une cellule est reconstruite mais où le morceau de surface correspondant n'intersecte pas le rayon peut se produire. Mais ce surcoût sera d'autant plus léger, que la profondeur de l'octree est importante et que ses feuilles sont fines. Cette technique va donc particulièrement bien se prêter aux nuages de points volumineux. Notre implémentation fournit une passerelle vers *Povray* [Pov04], qui a permis par exemple de générer l'image du centre de la figure 11.

## 6 Résultats

Le tableau 12 récapitule les temps de reconstruction de divers modèles de tailles variées. La figure 13 montre

| Modèles | Nb points | Part. | Rec. |
|---|---|---|---|
| Torse | 2 941 | 0.01 | 0.341 |
| Bouteille | 42 736 | 0.03 | 3.09 |
| AdamKraft | 70 073 | 0.52 | 6.09 |
| Robinet | 71 481 | 0.08 | 5.06 |
| Dragon | 100 251 | 0.20 | 7.03 |
| Elephant | 148 689 | 0.17 | 11.25 |
| Buddha | 144 648 | 0.25 | 12.70 |
| Dragon | 437 646 | 0.98 | 46.21 |
| Buddha | 530 000 | 1.02 | 47.60 |

FIG. 12 – Temps de génération (en sec.) des surfaces sur un Intel PIV 3.4 Ghz, 1.5 Go.

l'évolution du temps de génération de la collection de surfaces en fonction du nombre de points pour le modèle du Dragon. Comme prévu, la courbe présente une asymptote linéaire à partir d'une certaine taille. Les surfaces générées fournissent une bonne visualisation sur les modèles testés. La qualité visuelle obtenue (cf. figure 14) est tout à fait comparable aux techniques couplant une reconstruction implicite suivie d'une extraction de maillage.

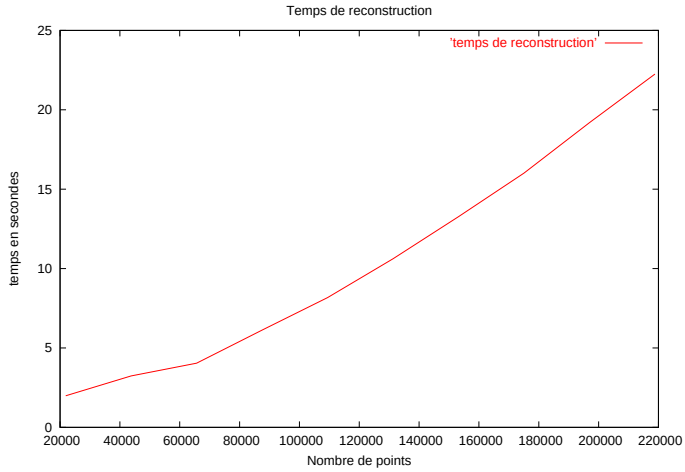


FIG. 13 – Temps de reconstruction du Dragon à diverses densités (rééchantillonnage par sélection aléatoire des points à partir du modèle haute résolution) : l'algorithme a une complexité linéaire.

Mais les temps de calculs résultants sont diminués d'un facteur pouvant aller jusqu'à 200 selon les cas. Les seules zones sensibles sont les zones où plusieurs topologies sont créées (ie. les zones de recouvrement). Il en résulte que dans les zones de fortes courbures, la densité doit être suffisante pour ne pas aboutir à des aberrations générées par l'opérateur de projection lors de la reconstruction locale. C'est en fait le critère de partitionnement qui est critique ici. Mais comme nous l'avons déjà signalé, la vitesse de reconstruction est telle qu'il est relativement facile d'ajuster les paramètres, intuitifs, du partitionnement en fonction des spécificités de l'objet traité.

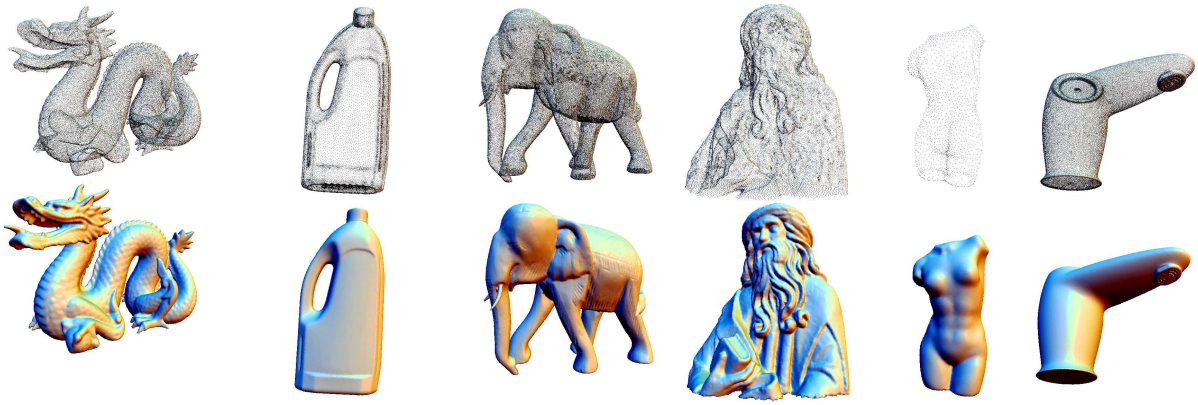

FIG. 14 – Reconstruction et visualisation de divers nuages de points (temps présents dans le tableau 12).

# 7 Conclusion et Travaux Futurs

L'étude menée a dévoilé plusieurs aspects de la reconstruction explicite de surfaces de points pour la visualisation :
- des algorithmes rapides peuvent être mis-en-oeuvre en adoptant une approche locale ;
- garantir une 2-variété unique pour l'ensemble de la surface n'est pas nécessaire en terme de visualisation : une surface peut tout-à-fait apparaître *visuellement continue* (même avec des zooms importants sur des détails de la surface) alors qu'elle ne l'est pas géométriquement ;
- les techniques de subdivision peuvent alors aider à obtenir cette notion de continuité visuelle ;
- il est possible d'appliquer une approche comparable à la *partition de l'unité* aux surfaces polygonales via la subdivision.

Nous avons introduit une reconstruction locale de surfaces de subdivision. Les résultats montrent que cette approche est viable en terme de visualisation. Les techniques originales proposées se présentent comme une composante d'un panel général de modélisation par points, et comme un complément aux méthodes de haute qualité mais coûteuses, complexes et difficiles à visualiser telles que celles basées sur les surfaces implicites. La performance du système développé permet de régler certains paramètres interactivement, et de visualiser en temps réel des surfaces de points sans avoir à modifier le pipeline ni les algorithmes de rendu classique, ce que ne permet pas les approches de type *splatting*.

Cette approche est particulièrement bien adaptée aux nuages de points denses. Les premiers résultats obtenus sont encourageant et ouvrent de nouvelles perspectives :
– formalisation d'un schéma de subdivision pour conformer des surfaces entre elles ;
– mise à profit de la multirésolution : modification de l'algorithme de manière à s'adapter à une transmission progressive des données, un des nombreux avantages des surfaces de points ;
– mise à profit de la localité : modification de l'algorithme de manière à permettre une reconstruction hors-mémoire (ni le nuage de points, ni la surface reconstruite ne sont présents dans leur intégralité en mémoire à un moment donné), qui pourra notamment être porté sur des grappes de PC à l'aide d'un support exécutif parallèle.

**Remerciements** Nous tenons à remercier le Stanford Computer Graphics Laboratory ainsi que la société Polygon Technology pour la mise à disposition des différents objets présents dans cet article.

# Références

[AB01] Dominique Attali and Jean-Daniel Boissonnat. Complexity of the delaunay triangulation of points on polyhedral surfaces. Technical report, INRIA, 2001.

[ABCO+01] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Point set surfaces. *IEEE Visualization 2001*, pages 21–28, October 2001. ISBN 0-7803-7200-x.

[ACK01] Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust. In *Proceedings of the sixth ACM symposium on Solid modeling and applications*, pages 249–266. ACM Press, 2001.

[Alg95] Maria-Elena Algorri. *G´en´eration et simplification du maillage pour la reconstruction de surfaces de points non structur´es.* PhD thesis, ENST, 1995.

[BC00] Jean-Daniel Boissonnat and Frédéric Cazals. Smooth surface reconstruction via natural neighbour interpolation of distance functions. In *Proceedings of the sixteenth annual symposium on Computational geometry*, pages 223–232. ACM Press, 2000.

[BI98] Andrew Blake and Michael Isard. *Active Contours.* Springer-Verlag, 1998.

[Blo88] Jules Bloomenthal. Polygonization of implicit surfaces. *Computer Aided Geometric Design*, 5 :341–355, 1988.

[BLZ00] Henning Biermann, Adi Levin, and Denis Zorin. Piecewise smooth subdivision surfaces with normal control. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 113–120. ACM Press/Addison-Wesley Publishing Co., 2000.

[Bri02] David Brickhill. Practical implementation techniques for multi-resolution subdivision surfaces. Technical report, The Golden Gate Game Company, 2002.

[BS02] Jeffrey Bolz and Peter Schröder. Rapid evaluation of catmull-clark subdivision surfaces. In *Proceeding of the seventh international conference on 3D Web technology*, pages 11–17. ACM Press, 2002.

[BS03] Jeff Bolz and Peter Schröder. Evaluation of subdivision surfaces on programmable graphics hardware, 2003.

[BSK04] Mario Botsch, Michael Spernat, and Leif Kobbelt. Phong splatting. In *Symp. Point-Based Graphics*, 2004.

[COCS00] D. Cohen-Or, Y. Chrysanthou, and C. Silva. A survey of visibility for walkthrough applications, 2000.

[CSD02] David Cohen-Steiner and Frank Da. A greedy delaunay based surface reconstruction algorithm. Technical report, INRIA- Sophia Antipolis, 2002.

[DGH01] Tamal K. Dey, Joachim Giesen, and James Hudson. Delaunay based shape reconstruction from large data. In *Proceedings of the IEEE 2001 symposium on parallel and large-data visualization and graphics*, pages 19–27. IEEE Press, 2001.

[DQ01] Ye Duan and Hong Qin. Intelligent balloon : a subdivision-based deformable model for surface reconstruction of arbitrary topology. In *Proceedings of the sixth ACM symposium on Solid modeling and applications*, pages 47–58. ACM Press, 2001.

[DWBS03] Andreas Dietrich, Ingo Wald, Carsten Benthin, and Philipp Slusallek. The OpenRT Application Programming Interface – Towards A Common API for Interactive Ray Tracing. In *Proceedings of the 2003 OpenSG Symposium*, pages 23–31, Darmstadt, Germany, 2003. Eurographics Association.

[DYQS04] Ye Duan, Liu Yang, Hong Qin, and Dimitris Samaras. Shape reconstruction from 3d and 2d data using pde-based deformable surfaces. 2004.

[GK00] M. Gopi and S. Krishnan. A fast and efficient projection based approach for surface reconstruction, 2000.

[GKS00] M. Gopi, S. Krishnan, and C.T. Silva. Surface reconstruction based on lower dimensional localized delaunay triangulation. In *Proceedings of Eurographics*, volume 19, 2000.

[HBB01] Denis Zorin Henning Biermann, Ioana M. Martin and Fausto Bernardini. Sharp features on multiresolution subdivision surfaces. In *Proceedings of Pacific Graphics 2001*, 2001.

[HDD+92] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics*, 26(2) :71–78, 1992.

[HDD+93] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. *Computer Graphics*, 27(Annual Conference Series) :19–26, 1993.

[HDD+94] Hugues Hoppe, Tony DeRose, Tom Duchamp, Mark Halstead, Hubert Jin, John McDonald, Jean Schweitzer, and Werner Stuetzle. Piecewise smooth surface reconstruction. *Computer Graphics*, 28(Annual Conference Series) :295–302, 1994.

[HSK99] S. Takeuchi H. Suzuki and T. Kanai. Subdivision surface fitting to a range of points. In *Proceedings of Pacific Graphics 99*, 1999.

[JK02] W. Jeong and C. Kim. Direct reconstruction of displaced subdivision surface from unorganized points. pages 160–168, 2002.

[Kri03] Jaroslav Krivanek. Representing and rendering surfaces with points. Technical Report DC-PSR-2003-03, Department of Computer Science and Engineering, Czech Technical University in Prague, 2003.

[KWT87] M. Kass, A. Witkin, and D. Terzopoulos. Snakes : Active contour models. *Computer Vision*, 1 :321–331, 1987.

[LC87] W.E. Lorensen and H.E. Cline. Marching cubes : a high resolution 3d surfaceconstruction algorithm. *Computer Graphics*, 21 :163–169, 1987.

[Loo87] Charles Loop. Smooth subdivisions surfaces based on triangles. Master's thesis, Department of Mathematica, University of Utah, August 1987.

[LPC+00] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, Dave Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. The digital michelangelo project : 3d scanning of large statues. In *Proc. SIGGRAPH 2000*, 2000.

[OBA+03] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H. Seidel. Multi-level partition of unity implicits, 2003.

[Pov04] Povray. Povray, 2004. http ://www.povray.org.

[Sil03] Claudio T. Silva. Out-of-core algorithms for scientific visualization and computer graphics. IEEE Visualization 2003, Tutorial No. 4, October 2003.

[TO02] Greg Turk and James F. O'Brien. Implicit surfaces that interpolate. In *SMI*, 2002.

[TRS04] Ireneusz Tobor, Patrick Reuter, and Christophe Schlick. Efficient reconstruction of large scattered geometric datasets using the partition of unity and radial basis functions. In *WSCG (Winter School of Computer Graphics)*, feb 2004.

[VPBM01] Alex Vlachos, Jorg Peters, Chas Boyd, and Jason L. Mitchell. Curved pn triangles. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 159–166. ACM Press, 2001.

[ZPBG01] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Surface splatting. In Eugene Fiume, editor, *SIGGRAPH 2001, Computer Graphics Proceedings*, pages 371–378. ACM Press / ACM SIGGRAPH, 2001.

[ZPKG02] Matthias Zwicker, Mark Pauly, Oliver Knoll, and Markus Gross. Pointshop 3d : An interactive system for point-based surface editing, 2002.

[ZS00] Denis Zorin and Peter Schröder. Subdivision for modeling and animation. In *SIGGRAPH Courses Notes*, 2000.